Start of Lecture 4: DOMAINS: Scripts – Human Behaviour

3.4. Scripts

Definition: Scripts.

- *A script is plan of action.*
- *By a domain script we shall, more specifically, understand*
  - *the structured, almost, if not outright,*
  - *formally expressed, wording of a set of*
  - *rules and regulations.*
- See also
  - *license* and
  - *contract*.

  Definitions follow.

3.4.1. Routes as Scripts

3.4.1.1. Paths

92. A path is a triple:
    (a) a hub identifier, h_i, a link identifier, l_j, and another hub identifier, h_k, distinct from h_i,
    (b) such that there is a link ℓ with identifier l_j in a net n such that $\{h_i, h_k\}$ are the hub identifiers that can be observed from ℓ.

type
92. Pth = HI × LI × HI
axiom
92(a). ∀ (hi,li,hi′):Pth · ∃ n:N,l:L · l ∈ obs_Ls(n) ⇒
92(b). obs_LI(l)=li ∧ obs_HIs(l)={hi,hi′}

93. From a net one can extract all its paths:
    (a) if l is a link of the net,
    (b) l_j its identifier,
    (c) $\{h_i, h_k\}$ the identifiers of its connected hubs,
    (d) then (h_i, l_j, h_k) and (h_k, l_j, h_j) are paths of the net.

value
93. paths: N → Pth-**set**
93(a). paths(n) ≡
93(d). {(hi,lj,hk),(hk,lj,hi)|l:L,lj:LI,hi,hk:HI·l ∈ obs_Ls(n) ∧
93(b). lj=obs_LI(l) ∧
93(c). {hi,hk}=obs_HIs(l)}

94. From a net descriptor one can (likewise) extract all its paths:
    (a) Let h_i, h_k be any two distinct hub identifiers of the net descriptor (definition set),
    (b) such that they both map into a link identifier l_j,
    (c) then (h_i, l_j, h_k) and (h_k, l_j, h_j) are paths of the net.

value
93. paths: ND → Pth-**set**
93. paths(nd) ≡
94(a). {(hi,lj,hk),(hk,lj,hi)|hi,hk:HI,lj:LI · hi≠hk ∧ {hi,hk}⊆**dom** nd ⇒
94(b). lj ∈ **dom** nd(hi)∩ **dom** nd(hk)}

3.4.1.2. Routes

95. A route of a net is a sequence of zero, one or more paths such that
    (a) all paths of a route are paths of the net and
    (b) adjacent paths in the sequence "share" hub identifiers.

type
95. R = Pth*
axiom
95. ∀ r:R, ∃ n:N ·
95(a). **elems** r ⊆ paths(n) ∧
95(b). ∀ i:**Nat** · {i,i+1}⊆**inds** r ⇒
95(b). **let** (_,_,hi)=r(i), (hi′,_,_)=r(i+1) **in** hi=hi′ **end**

96. From a net, n, we can generate the possibly infinite set of finite and possibly infinite routes:
    (a) <> is a path (***basis clause 1***);
    (b) if p is a path of n then $< p >$ is a path of n (***basis clause 2***);
    (c) if r and r' are non-empty routes of n
        i. and the last h_i of r is the same as the first h_j of r'
        ii. then the concatenation of r and r' is a route (***induction clause***).
    (d) Only such routes which can be formed by a (finite, respectively infinite) application of basis clauses Items 96(a) and 96(b) and induction clause Item 96(c) are routes (***extremal clause***).

value
96. routes: N|ND → R-**infset**
96. routes(nond) ≡
96(a). **let** rs = {⟨⟩} ∪
96(b). {⟨p⟩|p:Pth·p ∈ paths(nond)} ∪
96((c))ii. {r⌢r′|r,r′:R · r ∈ rs ∧ r′ ∈ rs ∧
96((c))i. ∃ hi,hi′,hi″,hi‴:H,li:LI ·
96((c))i. r=r″⌢⟨(hi,li,hi′)⟩∧r′=⟨(hi″,li′,hi‴)⟩⌢r‴ ∧
96((c))i. hi′=hi″} **in**
96(d). rs **end**

3.4.2. Bus Timetables as Scripts

3.4.2.1. Buses

97. Buses are vehicles,
98. with bus identifiers being the same as vehicle identifiers.

type
97. B
98. BI ⊆ VI

3.4.2.2. Bus Stops

99. A link bus stop indicates the link (by its identifier), the from and to hub identifiers, and the fraction "down the link" from the from to the to hub identifiers.

type
99. BS = mkL_BS(sel_fhi:HI,sel_li:LI,sel_f:F,sel_thi:HI)

3.4.2.3. Bus Routes

100. A bus stop list is a sequence of two or more bus stops, *bsl*.
101. A bus route, *br*, is a pair of a net route, *r*, and a bus stop list , *bsl*, such that route *r* is a route of *n* and such that *bsl* is embedded in *r*. If
    (a) there exists an index list, *il*, of ascending indices of the route *r* and of the length of *bsl*
    (b) such that the *i*th path of *r*
    (c) share from and to hub identifiers and link identifier with the *il*(*i*)th bus stop of *bsl*

    then *bsl* is embedded in *r*.
102. We must allow for two or more stops along a bus route to be adjacent on the same link — in which case the corresponding fractions must likewise be ascending.


```
value
  n:N
type
100.    BSL = BS*
101.    BR = {|(r,bsl):(R×BSL)•wf_BR(r,bsl)|}
value
101.    wf_BR: BR → Bool
101.    wf_BR(r,bsl) ≡ ∃ n:N,r:R•r ∈ routes(n) ∧ is_embedded_in(r,bsl)

101(a).   is_embedded_in: BR → Bool
101(a).   is_embedded_in(r,bsl) ≡
101(b).     ∃ il:Nat* • len il=len bsl∧inds il⊆inds r∧ascending(il) ⇒
101(c).       ∀ i:Nat • i ∈ inds il ⇒
101(c).         let (hi,lj,hk) = r(il(i)),(hi′,lj′,f,hk′) = bsl(i) in
101(c).         hi=hi′ ∧ lj=lj′ ∧ hk=hk′ end ∧
102.       ∀ i:Nat • {i,i+1}⊆inds il ⇒
102.         let (hi,lj,f,hk)=bsl(i),(hi′,lj′,f′,hk′)=bsl(i+1) in
102.         hi=hi′ ∧ lj=lj′ ∧ hk=hk′ ⇒ f<f′ end

    ascending: Nat* → Bool, ascending(il) ≡ ∀ i:Nat•{i,i+1}⊆inds il ⇒ il(i)≤il(i+1)
```


3.4.2.4. Bus Schedule

103. A timed bus stop is a pair of a time and a bus stop.
104. A timed bus stop list is a sequence of timed bus stops.
105. A bus schedule is a pair of a route and a timed bus stop list such that
    - there is a net of which the routes is indeed a route,
    - the bus stop list of the timed bus stop list is embedded in the route, and
    - 'later" listed bus stops register later times.
106. SimpleBusSchedules remove routes from BusRoutes.

type
103. TBS :: sel_T:T sel_bs:BS
104. TBSL = TBS*
105. BusSched = {|(r,tbsl):(R×TBSL)•wf_BusSched(r,tbsl)|}
value
105. wf_BusSched: BusSched → **Bool**
105. wf_BusSched(r,tbsl) ≡
105. ∃ n:N•r ∈ routes(n)
105. ∧ **let** bsl:SBS = ⟨sel_BS(tbsl(i))|i:[1..**len** tbsl]⟩ **in** is_embedded_in(r,bsl) **end**
105. ∧ ∀ i:**Nat**•{i,i+1}⊆**inds** tbsl ⇒ sel_T(tbsl(i))<sel_T(tbsl(i+1))
type
106. SBS = {|bsl:BS*•∃ n:N,r:R•r ∈ routes(n)∧is_embedded_in(r,bsl)|}

3.4.2.5. Timetable

The concept of a bus line captures all those bus schedules which ply the same bus route but at different times. A timetable is made up from distinctly named bus lines.

107. A bus line has a unique bus line name.
108. We say that two bus schedules are the same if they are based on the same route and if they differ only in their times.
109. Each of the different bus routes of a bus line has a unique bus number.
110. A route bus schedule pairs a route with simple bus schedules for each of a number of busses (identified by their bus number).
111. A bus timetable (listing, map) maps bus line names to route bus schedules.
112. A timetable is a pair, a net and a table.
113. A well-formed timetable must satisfy same bus schedules within each bus line
114. All bus numbers are distinct across bus lines.

type
107. BLNm
value
108. same_bus_schedule: BusSched × BusSched → **Bool**
108. same_bus_schedule((r1,btl1),(r2,btl2)) ≡
108. r1 = r2 ∧ **len** btl1 = btl2 ∧
108. ⟨sel_BS(btl1(i))|i:[1..**len** btl1]⟩=⟨sel_BS(btl2(i))|i:[1..**len** btl2]⟩
type
109. BNo
110. RBS :: sel_R:R sel_btbl:(BNo $\overrightarrow{m}$ SBS)
111. TBL = BLNm $\overrightarrow{m}$ RBS
112. TT′ = ND × TBL
113. TT = {|tt:TT′•wf_TT(tt)|}

value
113. wf_TT: TT′ → **Bool**
113. wf_TT(_,tbl) ≡
113. ∀ bln:BLNm•bln ∈ **dom** tbl ⇒
113. ∀ bno,bno′:BNo • {bno,bno′}⊆**dom** sel_btbl(tbl(bln)) ⇒
113. same_bus_schedule(sel_R(tbl(bln)),sel_btbl(tbl(bln))(bno),
113. sel_R(tbl(bln)),sel_btbl(tbl(bln))(bno′)) ∧
114. ∀ bln′,bln″:BLNm • {bln′,bln″}⊆**dom** tbl ∧ bln′≠bln″ ⇒
114. **dom** sel_btbl(tbl(bln′)) ∩ **dom** sel_btbl(tbl(bln″)) = {}

3.4.3. Route and Bus Timetable Denotations

- What are routes and bus timetables scripting ?
- Routes (list of connected link traversal designations) script that one may transport people or freight along the sequence of designated links.
- Bus timetables script (at least) two things:
  - the set of bus traffics on the net which satisfy the bus timetable, and
  - information that potential and actual bus passengers may, within some measure of statistics (and probability), rely upon for their bus transport.

- Here, we shall not develop the idea of bus timetables denoting certain traffics.
  - Instead we refer to our previously sketched model of traffics (Sect. , Pages 89–97).
- Route (designations) and bus timetables
  - script potential and actual route travels, respectively
  - script the dispatch of buses and their travelling.
- Bus timetables can also be seen as a form of contracts
  - between the bus operators offering the bus services
  - and potential and actual passengers,
  - with the contract promising timely transport.
- In the next section, Sect. , we shall sketch a language of bus service contracts and bus service actions implied by such contracts.

3.4.4. Licenses and Contracts

Definition: License.

- *A license is*
  - *a script*
  - *specifically expressing a permission to act;*
  - *is freedom of action;*
  - *is a permission granted by competent authority to engage in a business or occupation or in an activity otherwise unlawful;*
  - *a document, plate, or tag evidencing a license granted;*
  - *a grant by the holder of a copyright or patent to another of any of the rights embodied in the copyright or patent short of an assignment of all rights.*

Licenses appear more to have morally than legally binding poser.

Definition: Contract.

- *A contract*
  - *is a special kind of license*
  - *specifically expressing a legally binding agreement between two or more parties —*
  - *hence a document describing the conditions of the contract;*
  - *a contract is business arrangement for the supply of goods or services at fixed prices, times and locations.*

- *In software development a contract specifies what is to be developed:*
  - *(1) a* ***domain description****,*
  - *(2) a* ***requirements prescription****, or*
  - *(3) a* ***software design****;*

  or a combination of these (1–2, 2–3, 1–3).

- *A contract further specifies*
  - *how it might, or must be developed;*
  - *criteria for acceptance of what has been developed;*
  - *delivery dates for the developed items;*
  - *who the "parties" to the contract are:*
    - *the* ***client*** *and*
    - *the* ***developer****, etc.*

- For a comprehensive treatment of licenses and contracts we refer to [`Chapter 10, Sect. 10.6 (Pages 309--326) [jaist-db10]][jaist-mono]`.
- We shall illustrate fragments of a language for bus service contracts.
- The background for the bus contract language is the following.
  - In many large cities around Europe the city or provincial government secures public transport in the form of bus services operated by many different private companies.
  - Earlier lectures illustrated the concept of bus (service) timetables.
  - The bus services implied by such a timetable, for a city area — with surrounding suburbs etc. — need not be implemented by just one company, but can be contracted, by the city government public transport office, to several companies, each taking care of a subset of the timetable.

  - Different bus operators then take care of non-overlapping parts and all take care of the full timetable.
  - It may even be that extra buses need be scheduled, on the fly, in connection with major sports or concert or other events.
  - Bus operators may experience vehicle breakdowns or bus driver shortages and may be forced to subcontract other, even otherwise competing bus operators to "step in" and alleviate the problem.

3.4.4.1. Contracts

Schematically we may represent a bus contract as follows:

Contract cn **between contractee** ci **and contractor** cj:
This contract contracts cj **in the period** $[t,t']$ **to**
perform the following services with respect to timetable tt:
operate bus lines $\{blj_1,blj_2,...,blj_n\}$
subject to the following occasional exceptions:
cancellation of bus tours:
$\{(blj_a,\{bno_{a_1},...,bno_{a_m}\}),...\}$ **subject to conditions** cbt
insertion of bus tours on lines
$\{blj_\alpha,blj_\beta,...,blj_\gamma\}$ **subject to conditions** ibt
subcontracting bus tours on lines
$\{blj_\delta,blj_\phi,...,blj_\omega\}$ **subject to conditions** scbt.

115. A bus contract has a header with the distinct names of a contractee and a contractor and a time interval.
116. A bus contract presents a timetable.
117. A bus contract presents a set of bus lines (by their identifiers) such that these are in the timetable.
118. And a bus contract may list one or more of three kinds of "exceptions":
    (a) cancellation of one or more named bus tours on one or more bus lines subject to certain (specified) conditions;
    (b) insertion of one or more extra bus tours on one or more bus lines subject to certain (specified) conditions;
    (c) subcontracting one or more unspecified bus tours on one or more bus lines subject to certain (specified) conditions — to further unspecified contractors.

We abstract the above quoted "one or more of three kinds of exceptions" as one possibly empty clause for each of these alternatives.

119. A bus contract now contains a header, a timetable, the subject bus lines and the exceptions,
120. such that
    (a) line names mentioned in the contract are those of the bus lines of the timetable, and
    (b) bus (tour) numbers are those of the appropriate bus lines in the timetable.
121. The calendar period is for at least one full day, midnight to midnight.
122. A named contract is a pair of a contract name and a contract.

type
115. CNm, CId, D, T, CON
115. CH = CId × CId × (D×D)
116. CT = TT
117. CLs = BLNm-**set**
118. CE = (CA × IN × SC) × CON
118(a). CA = BLNm $\overrightarrow{m}$ BNo-**set**
118(b). IN = BLNm $\overrightarrow{m}$ BNo-**set**
118(c). SC = BLNm-**set**
119. CO′ = CH × CT × CLs × CE
120. CO = {| co:CO′•wf_CO(co) |}
122. NCO = CNm × CO

value

120. wf_CO: CO′ → **Bool**
120. wf_CO((ce,cr,(d,d′)),(nd,tbl),cls,((blns,blns′,bls),con)) ≡
117. ce ≠ cr ∧
120(a). cls ⊆ **dom** tbl ∧
120(b). ∀ bli,bli′:BLNm · bli ∈ **dom** blns ∧ bli′ ∈ **dom** blns′ ⇒
120(a). {bli,bli′} ⊆ **dom** tbl ∧
120(b). blns(bli) ∪ blns′(bli′) ⊆ **dom** sel_btbtl(tbl(bli)) ∧
120(a). bls ⊂ **dom** tbl ∧
121. d < d′

3.4.4.2. Contractual Actions

For contract cn **commence bus tour, line:** bli **and bus no.:** bno

For contract cn **cancel bus tour, line:** bli **and bus no.:** bno

For contract cn **insert extra bus tour, line:** bli **and bus no.:** bno

Subcontract with respect to contract cn **the following:**
Contract cn′: **for the calendar period** [d,d′] **contractee** ci **contracts contractor** cj
to perform the following services with respect to timetable tt:
operate bus lines $\{blj_1,blj_2,...,blj_n\}$
subject to the following occasional exceptions:
cancellation of bus tours:
$\{(blj_c,\{bno_{c_1},...,bno_{c_m}\}),...\}$ **subject to conditions** cbt
insertion of bus tours on lines
$\{(blj_i,\{bno_{i_1},...,bno_{i_n}\}),...\}$ **subject to conditions** ibt
subcontracting bus tours on lines
$\{blj_\delta,blj_\phi,...,blj_\omega\}$ **subject to conditions** scbt.

123. A bus operator action is either a commence, a cancellation, an insertion or a subcontracting action. All actions refer to the (name of) the contract with respect to which the action is contracted.
    (a) A `commence` action designator states the bus line concerned and the bus number of that line.
    (b) A cancellation action designator states the bus line concerned and the bus number of that line.
    (c) An insertion action designator states the bus line concerned and the bus number of that line — for which an extra bus is to be inserted.[4]
    (d) A subcontracting action designator, besides the name of the contract with respect to which the subcontract is a subcontract, state a named contract (whose contract name is unique).

[4] The insertion of buses in connection with either unscheduled or extraordinary (sports, concerts, etc.) events can be handled by special, initial contracts.

type

123. Act = Com | Can | Ins | Sub
123(a). Com == mkCom(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
123(b). Can == mkCan(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
123(c). Ins == mkIns(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
123(d). Sub == mkSub(sel_cn:CNm,sel_con:NCO)

3.4.4.3. Wellformedness of Contractual Actions

124. In order to express wellformedness conditions, that is, pre-conditions, for the action designators we introduce a context which map contract names to contracts.
125. Wellformedness of a contract is now expressed with respect to a context.

type
124. CTX = CNm $\overrightarrow{m}$ CO
value
125. wf_Act: Act → CTX → **Bool**

- Let a defined cnm entry in ctx be a contract: ((ce,cr),(nd,tbl),cls,(blns,bls,bls′),(d,d′)).

126. If cmd is a commence command mkCom(cnm,bln,bno), then
   (a) contract name cnm must be defined in context ctx;
   (b) bus line name bln must be defined in the contract, that is, in cls, and
   (c) bus number bno must be defined in the bus table part of table tbl.

126. wf_Act(mkCom(cnm,bln,bno))(ctx) ≡
126(a). cnm ∈ **dom** ctx ∧
126. **let** ((ce,cr),(nd,tbl),cls,(blns,bls,bls′),(d,d′)) = ctx(cnm) **in**
126(b). bln ∈ cls ∧
126(c). bno ∈ **dom** sel_btbl(tbl(bln)) **end**

127. cancellation and insertion commands have the same static wellformedness conditions as have commence command.

127. wf_Act(mkCan(cnm,bln,bno))(ctx) ≡ wf_Act(mkCom(cnm,bln,bno))(ctx)
127. wf_Act(mkIns(cnm,bln,bno))(ctx) ≡ wf_Act(mkCom(cnm,bln,bno))(ctx)

128. If cmd is a subcontract command then
   Let the subcontract command and the cnm named contract in ctx be
   mkSub(cnm,nco:(cnm′,(ce′,cr′,(d″,d‴)),(nd′,tbl′),cls′,(blns′,bls″,bls‴)))
   respectively ((ce,cr,(d,d′)), (nd,tbl), cls, (blns,bls,bls′)).
   (a) contract name cnm must be defined in context ctx;
   (b) contract name cnm′ must not be defined in context ctx;
   (c) the calendar period of the subcontract must be within that of the contract from which it derives;
   (d) the net descriptors nd and nd′ must be identical;
   (e) the tables tbl and tbl′ and must be identical and
   (f) the set, cls′, of bus line names that are the scope of the subcontracting must be a subset of bls′.

128. wf_Act(mkSub(cnm,nco:(cnm′,co:((ce′,cr′,(d″,d‴)),(nd′,tbl′),cls′,(blns′,blns″,bls‴)))))(ctx)
128(a). cnm ∈ **dom** ctx ∧
128. **let** co′ = ((ce,cr,(d,d′)),(nd,tbl),cls,(blns,blns′,bls′)) = ctx(cnm) **in**
128(b). cnm′ ∉ **dom** tbl ∧
128(c). d ≤ d″ ≤ d‴ ≤ d′ ∧
128(d). nd′ = nd ∧
128(e). tbl′ = tbl ∧
128(f). cls′ ⊆ bls′ **end**

- Wellformedness of contracts, wf_CO(co) and wf_CO(co′), secures other constraints.

- We do not here bring any narrated or formalised description of the semantics of contracts and actions.
- First such a description would be rather lengthy.
- Secondly a specification would be more of a requirements prescription.

3.5. Management and Organisation

Definition: Management.

- *Management is about resources:*
  - *their acquisition,*
  - *scheduling (over time),*
  - *allocation (over locations),*
  - *deployment (in performing actions) and*
  - *disposal (“retirement”).*
- *We distinguish between*
  - *board-directed,*
  - *strategic,*
  - *tactical and*
  - *operational*

  actions.

- *Board-directed actions target mainly financial resources: obtaining new funds through conversion of goodwill into financial resources, acquiring and selling “competing” or “supplementary” business units.*
- *Strategic actions convert financial resources into production, service supplies and resources and vice-versa — and in this these actions schedule availability of such resources.*
- *Tactical actions mainly allocate resources.*
- *Operational actions order, monitor and control the deployment of resources in the performance of actions.*

Definition: Organisation.

- *Organisation is about*
  - *the "grand scale",*
    - *executive and strategic*
    - *national, continental or global (world wide)*
  - *(i) allocation of major resource (e.g., business) units, whether in a hierarchical, in a matrix, or in some other organigram-specified structure,*
  - *(ii) as well as the clearly defined relations (which information, decisions and actions are transferred) between these units, and*
  - *(iii) organisational dynamics.*

Definition: Management & Organisation.

- *The composite term management and organisation*
  - *applies in connection with management as outlined just above and*
  - *with organisation also outlined above.*
- *The term then emphasises the relations between the organisation and management of an enterprise.*

• • •

The borderlines within management actions and across organisation "layouts" are fuzzy.

3.5.1. Transport System Examples

We shall only present sketchy examples of management and organsation.

- Executive actions:
  - Deciding on major re-organisation of a transport net
    - (for example introduction of toll roads or freeways,
    - road pricing,
    - major bridges across wide waters [potentially connecting two hitherto unconnected nets],
    - and their management)

    are executive actions.
  - So are decisions on merging or splitting transport from or into several transport services.

  - Reorganising an enterprise
    - from one characterised by a "deep" hierarchy of management layers (a hierarchy which may very well exemplify highly centralised both administrative and functional monitoring and control)
    - into a matrix of two "shallow" hierarchies, one which addresses tactical and operational management and one which addresses executive and strategic management — with the former (the operations) being replicated across geographical areas while the latter applies "globally" —

    such reorganisations reflect executive actions (but are carried out by strategic and tactical management).

- Strategic actions: Adding or removing transport links, or major reorganisation of bus timetables are strategic actions. Splitting a(n own) contract into what is still to be operated and subcontracting other parts, for definite, to other bus operators are also strategic actions.
- Tactical actions: Insertion and cancellation of bus services are tactical actions. Subcontracting some parts of a timetable demanded service, for a short while, to other bus operators could be considered tactical actions.
- Operational actions: Commencing and thus, in general, allocating drivers to and sending these off on bus services are operational actions. So are announcing insertion of new (unscheduled) and cancellation of scheduled routes.

3.6. Human Behaviour

Definition: Human Behaviour.

- *By human behaviour we shall here understand*
  - *the way a human follows the enterprise* rules and regulations
  - *as well as interacts with a* machine:
    - dutifully *honouring specified (machine* dialogue *or)* protocols*,*
    - *or* negligently *so,*
    - *or* sloppily *not quite so,*
    - *or even* criminally *not so!*
- *Human behaviour is a* facet *of the* domain*.*
  - *We shall thus model human behaviour also in terms of it failing to react properly,*
  - *i.e., humans as* non-deterministic agents*!*

3.7. Towards Theories of Domain Facets

3.7.1. A Theory of Intrinsics

3.7.2. Theories of Support Technologies

3.7.2.1. An Example

- Traffic (tf:TF), intrinsically, is a total function over some time interval, from time (t:T) to continuously positioned (p:P) vehicles (tn:TN).
- Conventional optical sensors sample, at regular intervals, the intrinsic train traffic.
- The result is a sampled traffic (stf:sTF).
- Hence the collection of all optical sensors, for any given net, is a partial function from intrinsic (itf) to sampled train traffics (stf).
- We need to express quality criteria that any optical sensor technology should satisfy — relative to a necessary and sufficient description of a closeness predicate.

- For all intrinsic traffics, itf, and for all optical sensor technologies, og, the following must hold:
  - Let stf be the traffic sampled by the optical gates.
  - For all time points, t, in the sampled traffic,
  - those time points must also be in the intrinsic traffic,
  - and, for all trains, tn, in the intrinsic traffic at that time,
  - the train must be observed by the optical gates, and
  - the actual position of the train and the sampled position must somehow be checkable to be close, or identical to one another.

Since hubs change state with time, n:N, the net needs to be part of any model of traffic.


```
type
  T, TN
  P = HP | LP
  NetTraffic :: net:N × trf:(V →m P)
  iTF = T → NetTraffic
  sTF = T →m NetTraffic
  oG = iTF ~→ sTF
value
  [close] c: NetTraffic × TN × NetTraffic ~→ Bool
axiom
  ∀ itt:iTF, og:OG · let stt = og(itt) in
    ∀ t:T · t ∈ dom stt ·
      t ∈ 𝔻𝕆𝕄 itt ∧ ∀ Tn:TN · tn ∈ dom trf(itt(t))
        ⇒ tn ∈ dom trf(stt(t)) ∧ c(itt(t),tn,stt(t)) end
```

3.7.2.2. General

- The formal requirements can be narrated:
  - Let Θ_i and Θ_a designate the spaces of intrinsic and actual-world configurations (contexts and states).
  - For each intrinsic configuration model — that we know is support technology assisted —
  - there exists a support technology solution,
  - that is, a total function from all intrinsic configurations to corresponding actual configurations.
- If we are not convinced that there is such a function then there is little hope that we can trust this technology

type
Θ_i, Θ_a
$\mathrm{ST} = \Theta_i \rightarrow \Theta_a$
axiom
$\forall\ \mathrm{sts:ST\text{-}\mathbf{set}},\ \mathrm{st:ST} \cdot \mathrm{st} \in \mathrm{sts} \Rightarrow \forall\ \theta_i{:}\Theta_i,\ \exists\ \theta_a{:}\Theta_a \cdot \mathrm{st}(\theta_i) = \theta_a$

3.7.3. A Theory of Rules & Regulations

- There are, abstractly speaking, usually three kinds of languages involved wrt. (i.e., when expressing) rules and regulations (respectively when invoking actions that are subject to rules and regulations).
  - Two languages, Rules and Reg, exist for describing rules, respectively regulations; and
  - one, Stimulus, exists for describing the form of the [always current] domain action stimuli.

- A syntactic stimulus, sy_sti, denotes a function, se_sti:STI: $\Theta \rightarrow \Theta$, from any configuration to a next configuration
- A syntactic rule, sy_rul:Rule, has as its semantics, its meaning, rul:RUL,
  - a predicate over current and next configurations, $(\Theta \times \Theta) \rightarrow$ **Bool**,
  - where these next configurations have been caused, by the stimuli. These stimuli express:
  - If the predicate holds then the stimulus will result in a valid next configuration.

type
Stimulus, Rule, Θ
$\mathrm{STI} = \Theta \rightarrow \Theta$
$\mathrm{RUL} = (\Theta \times \Theta) \rightarrow \mathbf{Bool}$
value
meaning: Stimulus $\rightarrow$ STI
meaning: Rule $\rightarrow$ RUL

valid: Stimulus $\times$ Rule $\rightarrow \Theta \rightarrow \mathbf{Bool}$
valid(sy_sti,sy_rul)(θ) $\equiv$ meaning(sy_rul)(θ,(meaning(sy_sti))(θ))

valid: Stimulus $\times$ RUL $\rightarrow \Theta \rightarrow \mathbf{Bool}$
valid(sy_sti,se_rul)(θ) $\equiv$ se_rul(θ,(meaning(sy_sti))(θ))

- A syntactic regulation, sy_reg:Reg (related to a specific rule), stands for, i.e., has as its semantics, its meaning,
  - a semantic regulation, se_reg:REG,
  - which is a pair.
  - This pair consists of
    - a predicate, pre_reg:Pre_REG, where Pre_REG $= (\Theta \times \Theta) \rightarrow$ **Bool**,
    - and a domain configuration-changing function, act_reg:Act_REG, where Act_REG $= \Theta \rightarrow \Theta$,
    - that is, both involving current and next domain configurations.

  - The two kinds of functions express:
    - If the predicate holds,
    - then the action can be applied.
- The predicate is almost the inverse of the rules functions.
- The action function serves to undo the stimulus function.

type
  Reg
  Rul_and_Reg = Rule × Reg
  REG = Pre_REG × Act_REG
  Pre_REG $= \Theta \times \Theta \rightarrow$ **Bool**
  Act_REG $= \Theta \rightarrow \Theta$
value
  interpret: Reg → REG

- The idea is now the following:
  - Any action of the system, i.e., the application of any stimulus,
    - may be an action in accordance with the rules,
    - or it may not.
  - Rules therefore express whether stimuli are valid or not in the current configuration.
  - And regulations therefore express whether they should be applied, and, if so, with what effort.

- More specifically,
  - there is usually, in any current system configuration, given a set of pairs of rules and regulations.
  - Let (sy_rul,sy_reg) be any such pair.
  - Let sy_sti be any possible stimulus.
  - And let θ be the current configuration.
  - Let the stimulus, sy_sti, applied in that configuration result in a next configuration, θ', where θ' = (meaning(sy_sti))(θ).
  - Let θ' (= (meaning(sy_sti))(θ)) violate the rule, i.e., ∼valid(sy_sti,sy_rul)(θ
  - then if predicate part, pre_reg, of the meaning of the regulation, sy_reg, holds in that violating next configuration, pre_reg(θ,θ'
  - then the action part, act_reg, of the meaning of the regulation, sy_reg, must be applied, act_reg(θ'), to remedy the situation.


```
axiom
 ∀ (sy_rul,sy_reg):Rul_and_Regs ·
  let se_rul = meaning(sy_rul),
      (pre_reg,act_reg) = meaning(sy_reg) in
  ∀ sy_sti:Stimulus, θ:Θ ·
   ∼valid(sy_sti,se_rul)(θ)
      ⇒ let θ′ = (meaning(sy_sti))(θ) in
        pre_reg(θ,θ′)
        ⇒ ∃ nθ:Θ · act_reg(θ′)=nθ ∧ se_rul(θ,nθ)
  end end
```


- It may be that the regulation predicate fails to detect applicability of regulations actions.
- That is, the interpretation of a rule differs, in that respect, from the interpretation of a regulation.
- Such is life in the domain, i.e., in actual reality

3.7.4. A Theory of Management & Organisation

3.7.5. A Theory of Human Behaviour

- Commensurate with the above, humans interpret rules and regulations differently,
- and not always "consistently" — in the sense of repeatedly applying the same interpretations.
- Our final specification pattern is therefore:

type
  Action = $\Theta \xrightarrow{\sim} \Theta$**-infset**
value
  hum_int: Rule $\to \Theta \to$ RUL**-infset**
  action: Stimulus $\to \Theta \to \Theta$
  hum_beha: Stimulus $\times$ Rules $\to$ Action $\to \Theta \xrightarrow{\sim} \Theta$**-infset**
  hum_beha(sy_sti,sy_rul)(α)(θ) **as** θset
   post
    θset = $\alpha(\theta)$ $\wedge$ action(sy_sti)(θ) $\in$ θset
    $\wedge$ $\forall$ θ':$\Theta\cdot\theta'$ $\in$ θset $\Rightarrow$
     $\exists$ se_rul:RUL$\cdot$se_rul $\in$ hum_int(sy_rul)(θ)$\Rightarrow$se_rul(θ,θ')

- The above is, necessarily, sketchy:
  - There is a possibly infinite variety of ways of interpreting some rules.
  - A human, in carrying out an action, interprets applicable rules and chooses one which that person believes suits some (professional, sloppy, delinquent or criminal) intent.
  - "Suits" means that it satisfies the intent,
    * i.e., yields **true** on the pre/post-configuration pair,
    * when the action is performed —
    * whether as intended by the ones who issued the rules and regulations or not.
  - We do not cover the case of whether an appropriate regulation is applied or not

End of Lecture 4: DOMAINS: Scripts – Human Behaviour

B Slide Table-of-Contents

Contents